PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-161985
(43)Date of publication of application : 19.06.1998

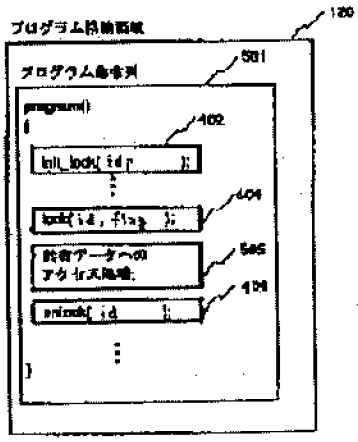
(51)Int.Cl. G06F 15/16
G06F 15/16
G06F 9/46
G06F 12/08

(21)Application number : 08-317496
(22)Date of filing : 28.11.1996

(71)Applicant : HITACHI LTD
(72)Inventor : YAMAUCHI MASAHIKO
YASHIRO HIROSHI
MURAYAMA HIDEKI
HORIKAWA KAZUO
HAYASHI TAKEHISA
YAMADA KIMITOSHI

(54) PROCESSOR ALLOCATING METHOD, AND MULTIPROCESSOR COMPUTER SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To eliminate useless data transfer between caches and to improve the execution efficiency of a computer system as a whole.
SOLUTION: An instruction string 505 for accessing shared data is sandwiched by a lock acquiring procedure 404 and a lock releasing procedure 406 and any processor is designated by a 2nd argument flag of the lock acquiring procedure 404. A multiprocessor computer system allocates a process for executing the instruction string 505 sandwiched by the lock acquiring procedure 404 and the lock releasing procedure 406 to the processor designated by the 2nd argument flag of the lock acquiring procedure 404. Since the plural processes for accessing the shared data are allocated to the same processor, there is no useless data transfer between caches and the execution efficiency of the entire system can be improved.



LEGAL STATUS
[Date of request for examination]
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号

特開平10－161985

(43)公開日　平成10年(1998)６月19日

(51)Int.Cl.[6]		識別記号	FI	
Ｇ０６Ｆ	15/16	３５０	Ｇ０６Ｆ　15/16	３５０Ａ
		３８０		３８０Ｚ
	9/46	３６０	9/46	３６０Ｅ
	12/08	３１０	12/08	３１０Ｂ

審査請求　未請求　請求項の数３　ＯＬ　（全 12 頁）

(21)出願番号　特願平8－317496

(22)出願日　平成８年(1996)11月28日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　山内　雅彦
神奈川県川崎市幸区鹿島田890番地の12
株式会社日立製作所情報・通信開発本部内

(72)発明者　曇代　寛
神奈川県川崎市幸区鹿島田890番地の12
株式会社日立製作所情報・通信開発本部内

(72)発明者　村山　秀樹
神奈川県川崎市幸区鹿島田890番地の12
株式会社日立製作所情報・通信開発本部内

(74)代理人　弁理士　有近　紳志郎

最終頁に続く

(54)【発明の名称】　プロセッサ割付方法およびマルチプロセッサ計算機システム

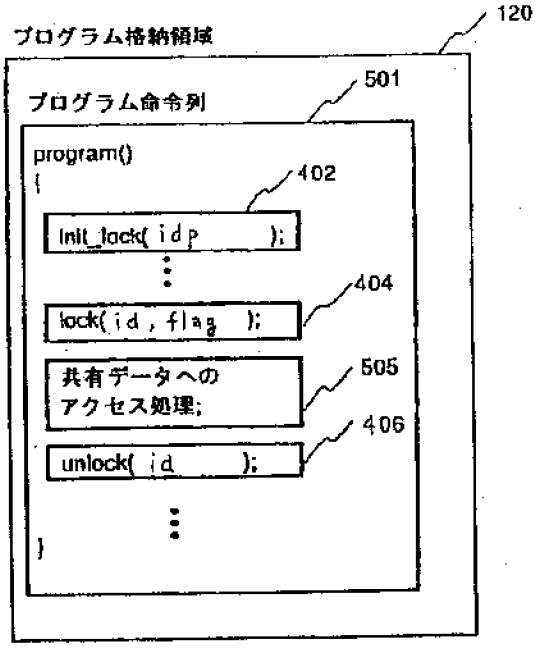
(57)【要約】

【課題】　キャッシュ間での無駄なデータ転送をなくし、システム全体の実行効率を向上させる。

【解決手段】　共有データにアクセスする命令列５０５をロック獲得手続き４０４とロック解放手続き４０６とで挟むと共にロック獲得手続き４０４の第2 引数flagでプロセッサを指定する。マルチプロセッサ計算機システムは、ロック獲得手続き４０４とロック解放手続き４０６とで挟まれた命令列５０５を実行するプロセスを、ロック獲得手続き４０４の第2 引数flagで指定したプロセッサに割り付ける。

【効果】　共有データにアクセスする複数のプロセスを同一のプロセッサに割り付けるため、キャッシュ間の無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。

図５



1

【特許請求の範囲】

【請求項1】　複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおけるプロセッサ割付方法であって、

複数のプロセスが共有データにアクセスするとき、それらプロセスを同一のプロセッサに割り付けることを特徴とするプロセッサ割付方法。

【請求項2】　複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおけるプロセッサ割付方法であって、

ある命令列の前に、プロセッサを指定しうる相互排除開始命令を配置し、前記命令列の後に、プロセッサの指定を解除しうる相互排除終了命令を配置し、前記相互排除開始命令でプロセッサが指定されている場合、その指定されたプロセッサにより前記相互排除開始命令と前記相互排除終了命令で挟まれた前記命令列を実行し、前記相互排除開始命令でプロセッサが指定されていない場合、任意のプロセッサにより前記相互排除開始命令と前記相互排除終了命令で挟まれた前記命令列を実行することを特徴とするプロセッサ割付方法。

【請求項3】　複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおいて、

相互排除開始命令によりプロセッサが指定されている場合は、その指定されたプロセッサにより、前記相互排除開始命令から後の命令を実行させ、相互排除終了命令を実行すると、任意のプロセッサにより、前記相互排除終了命令から後の命令を実行させるプロセッサ割付制御手段を具備することを特徴とするマルチプロセッサ計算機システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、プロセッサ割付方法およびマルチプロセッサ計算機システムに関し、更に詳しくは、キャッシュ間のデータ転送回数を減らし、システム全体の実行効率を向上させることができるプロセッサ割付方法およびマルチプロセッサ計算機システムに関する。

【０００２】

【従来の技術】複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおいて、異なるプロセッサに割り付けられた複数のプロセスが共有データにアクセスするとき、共有データの一貫性を保つため、前記異なるプロセッサのキャッシュ間で共有データを移動させる制御が行われている。かかる制御の詳細は「ＵＮＩＸカーネル内部解析　キャッシュとマルチプロセッサの管理、Ｃ.Schimmel著、前川守監訳、岩本信一訳、ソフトバンク株式会社、２９３～３２７頁」に記載されている。

2

【０００３】

【発明が解決しようとする課題】異なるプロセッサのキャッシュ間で共有データを移動させる制御を行う場合、例えば、異なるプロセッサに割り付けられた各プロセスが交互に共有データに書き込み動作を行なうと、キャッシュ間のデータ転送回数が増え、計算機システム全体の実行効率が下がる問題点がある。そこで、本発明の目的は、キャッシュ間のデータ転送回数を減らし、システム全体の実行効率を向上させることができるプロセッサ割付方法およびマルチプロセッサ計算機システムを提供することにある。

【０００４】

【課題を解決するための手段】第1の観点では、本発明は、複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおけるプロセッサ割付方法であって、複数のプロセスが共有データにアクセスするとき、それらプロセスを同一のプロセッサに割り付けることを特徴とするプロセッサ割付方法を提供する。上記プロセッサ割付方法では、共有データにアクセスする複数のプロセスを同一のプロセッサに割り付けるため、キャッシュ間の無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。また、複数の共有データがある場合には、各共有データごとに別々のプロセッサを対応させれば、負荷を分散することができる。

【０００５】第2の観点では、本発明は、複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおけるプロセッサ割付方法であって、ある命令列の前に、プロセッサを指定しうる相互排除開始命令を配置し、前記命令列の後に、プロセッサの指定を解除しうる相互排除終了命令を配置し、前記相互排除開始命令でプロセッサが指定されている場合、その指定されたプロセッサにより前記相互排除開始命令と前記相互排除終了命令で挟まれた前記命令列を実行し、前記相互排除開始命令でプロセッサが指定されていない場合、任意のプロセッサにより前記相互排除開始命令と前記相互排除終了命令で挟まれた前記命令列を実行することを特徴とするプロセッサ割付方法を提供する。上記プロセッサ割付方法では、共有データにアクセスする命令列を相互排除開始命令と相互排除終了命令とで挟むと共に相互排除開始命令でプロセッサを指定することにより、共有データにアクセスするプロセスを同一のプロセッサに割り付けることが出来る。このため、キャッシュ間の無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。また、共有データにアクセスする命令列が複数ある場合には、各命令列ごとに別々のプロセッサを指定すれば、負荷を分散することができる。なお、上記構成において、プロセッサの指定は、プロセッサ番号を明示して

3

行ってもよいし、マルチプロセッサ計算機システムに一つのプロセッサを選択させることにより行ってもよい。後者の場合、マルチプロセッサ計算機システムは、例えば、相互排除開始命令を実行したプロセッサの番号に基づいてプロセッサを選択したり、共有データが現時点で存在しているキャッシュに対応するプロセッサを選択する。

【０００６】第3の観点では、本発明は、複数のプロセッサとそれらプロセッサにそれぞれ対応するキャッシュとを有するマルチプロセッサ計算機システムにおいて、相互排除開始命令によりプロセッサが指定されている場合は、その指定されたプロセッサにより、前記相互排除開始命令から後の命令を実行させ、相互排除終了命令を実行すると、任意のプロセッサにより、前記相互排除終了命令から後の命令を実行させるプロセッサ割付制御手段を具備することを特徴とするマルチプロセッサ計算機システムを提供する。上記マルチプロセッサ計算機システムでは、プログラムにおいて共有データにアクセスする命令列を相互排除開始命令と相互排除終了命令とで挟むと共に相互排除開始命令でプロセッサを指定することにより、共有データにアクセスするプロセスを同一のプロセッサに割り付けることが出来る。このため、キャッシュ間の無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。また、共有データにアクセスする命令列が複数ある場合には、各命令列ごとに別々のプロセッサを指定すれば、負荷を分散することができる。なお、上記構成において、プロセッサの指定は、プロセッサ番号を明示して行ってもよいし、マルチプロセッサ計算機システムに一つのプロセッサを選択させることにより行ってもよい。後者の場合、プロセッサ割付制御手段は、例えば、相互排除開始命令を実行したプロセッサの番号に基づいてプロセッサを選択したり、共有データが現時点で存在しているキャッシュに対応するプロセッサを選択する。

【０００７】

【発明の実施の形態】図1は、本発明の一実施形態にかかるマルチプロセッサ計算機システム１０１のブロック図である。このマルチプロセッサ計算機システム１０１は、複数のプロセッサ１０２（１）～１０２（Ｎ）と、各プロセッサごとのキャッシュメモリ１０３（１）～１０３（Ｎ）と、命令列やデータを格納するための磁気ディスク等の2次記憶装置１０４と、メインメモリ１０６と、システム・バス１０５とを具備して構成されている。

【０００８】前記メインメモリ１０６には、プロセッサ１０２が実行するプログラム格納領域１２０（１）～（Ｌ）と、プロセスを管理するためのプロセス管理表１１７と、プロセッサ１０２（１）～１０２（Ｎ）で実行中のプロセスを管理するための実行中プロセス管理表１１６と、プロセッサ１０２（１）～１０２（Ｎ）には未だ割り付けられていないが実行可能な状態にあるプロセスをリスト構造で管理する実行可能プロセスリスト１１８と、プロセッサ１０２（１）～１０２（Ｎ）に割り付けるプロセスを切り替える手続を格納するプロセス切り替え手続き格納領域１１９と、複数のプロセスで共有するデータの一貫性を保つために他のプロセスが当該共有データを読み書きすることを禁止する相互排除の際に使用する情報を格納するためのロック用データ格納領域１１０（１）～（Ｍ）と、それらロック用データ格納領域１１０（１）～（Ｍ）を初期化する手続を格納するロック初期化手続き格納領域１０７と、共有データにアクセスする命令列の前で相互排除を指示する手続を格納するロック獲得手続き格納領域１０８と、共有データにアクセスする命令列の後で相互排除の解除を指示する手続を格納するロック解放手続き格納領域１０９とがある。

4

【０００９】前記ロック用データ格納領域１１０（１）～（Ｍ）には、対応するデータが相互排除中か否かを示すロック用変数格納領域１１１と、対応するデータにアクセス中のプロセスがあるか否かを示すロック獲得フラグ格納領域１１２と、対応するデータにアクセスするプロセスが割り付けられたプロセッサの番号を格納する割付プロセッサ番号格納領域１１３と、対応するデータにアクセスできる時を待っているプロセスをリストで管理している待機プロセスリスト１１４とがある。

【００１０】図2は、前記プロセス管理表１１７の構成図である。プロセス管理表１１７の各行は、それぞれ一つのプロセスに対応している。各行は、行番号を格納するインデックス２０１と、当該行に対応するプロセスの番号を格納するプロセス番号２０２と、当該行に対応するプロセスの状態を格納する実行状態２０３と、当該行に対応するプロセスの実行を中断した時にプログラムカウンタ等のレジスタ情報を退避した記憶領域を格納するレジスタ退避領域２０４と、実行可能プロセスリスト１１８のようなリスト構造を作るために使用するリンク先プロセスの行番号（またはリスト構造の末尾を示す識別子ＥＮＤ）を格納する次プロセスインデックス２０５と、当該行に対応するプロセスを割り付けるプロセッサが決まっている場合にその番号を格納する割付プロセッサ番号２０６とから構成される。

【００１１】図3は、前記実行中プロセス管理表１１６の構成図である。実行中プロセス管理表１１６の各行は、それぞれプロセッサ１０２（１）～１０２（Ｎ）に対応している。各行は、当該行に対応するプロセッサ１０２（１）～１０２（Ｎ）の番号（１～Ｎ）を格納するプロセッサ番号３０１と、当該行に対応するプロセッサ１０２が実行しているプロセスの番号を格納するプロセス番号３０２とから構成される。

【００１２】図4は、相互排除を実行するためにオペレーティング・システムが呼び出す手続きであるロック初期化４０２と、ロック獲得４０４と、ロック解放４０６の説明図である。

5

【００１３】ロック初期化(手続き名init__lock) ４０２は、ロック用データ識別子i ｄを返すための変数格納領域のアドレスi ｄｐを引数とする。そして、ロック用データ格納領域１１０を初期化し、その初期化したロック用データ格納領域１１０に与えたロック用データ識別子i ｄを前記アドレスi ｄｐが指す変数格納領域に格納する機能を有する。

【００１４】ロック獲得(手続き名lock) ４０４は、前記ロック初期化４０２で得たロック用データ識別子i ｄを第1 引数とし、機能の種類を指定する識別子flagを第２引数とする。そして、ロック用データ識別子i ｄに対応するデータに以後アクセスするプロセスを、第2 引数の識別子flagの値がプロセッサ番号(１～N) なら当該プロセッサ番号(１～N) に割り付け、識別子flagの値が－1 ならシステム側で選択した一つのプロセッサに割り付け、さらに、識別子flagの値が－2 なら従来技術で提供されているセマフォと同様な相互排除とプロセススケジューリング(プロセッサに割り付けるプロセスを選択する処理) を提供するように制御する機能を有する。なお、セマフォは、Ｐ命令とＶ命令とを使って相互排除を実現する。Ｐ命令とＶ命令の詳細は「オペレーティング・システムの機能と構成、高橋他、岩波書店、１９８３、１４７～１５０頁」や「UNIXカーネルの設計、Maurice 著／坂本他訳、共立出版、１９９０、３３４～３４１頁」に記載されている。また、プロセススケジューリングの詳細は「UNIXカーネルの設計、Maurice 著、坂本他訳、共立出版、１９９０、２１１～２１９頁」に記載されている。

【００１５】ロック解放(手続き名unlock) ４０６は、前記ロック初期化４０２で得たロック用データ識別子i ｄを引数とする。そして、ロック用データ識別子i ｄに対応するデータに以後アクセスするプロセスを任意のプロセッサに割り付けることを可能にする機能を有する。

【００１６】図5 は、ロック初期化４０２、ロック獲得４０４、ロック解放４０６の各手続きを含むプログラム命令列の例示図である。共有データにアクセスする複数のプロセス間で相互排除を行うべき命令列５０５の前にロック獲得４０４を置き、後にロック解放４０６を置く。そして、ロック獲得４０４より前にロック初期化４０２を置く。なお、ロック獲得４０４とロック解放４０６に挟まれた命令列５０５を危険領域という。

【００１７】図6 は、ロック初期化４０２の処理を示すＰＡＤ図である。ステップ６０１では、ロック用データ格納領域１１０を確保し、これを構造体lock__dataとする。ロック用データ格納領域１１０のロック用変数格納領域１１１、ロック獲得フラグ格納領域１１２、割付プロセッサ番号格納領域１１３および待機プロセスリスト１１４は、それぞれ構造体lock__dataのメンバlock、anybody、processor、listとなる。ステップ６０２では、構造体lock__dataのメンバlock、anybody、processor、

6

listに初期値を設定する。メンバlockの初期値FALSEは、対応するデータが相互排除されていないことを意味する。メンバanybody の初期値NO_PROCESSは、対応するデータにアクセス中のプロセスがないことを意味する。メンバprocessor の初期値－1 は、対応するデータにアクセスするプロセスがプロセッサに割り付けられていないことを意味する。メンバlistの初期値は、空(待機プロセスがない) を意味する識別子とする。ステップ６０３では、ロック用データ構造体lock__dataにロック用データ識別子i ｄを与える。ステップ６０４では、引数のアドレスi ｄｐが指す変数格納領域にロック用データ識別子i ｄを格納する。ステップ６０５では、ロック初期化４０２の呼び出し元に戻る。

【００１８】図7 は、ロック獲得４０４の処理を示すＰＡＤ図である。ステップ７０１では、第1 引数のロック用データ識別子i ｄに対応するロック用データ構造体lock__dataを得る。ステップ７０２では、第2 引数のフラグflagの値を判定し、値が1 ～N (プロセッサ番号) か又は－1 (プロセスを一定のプロセッサに割り付ける識別子) ならステップ７０３へ進み、値が－2 ならステップ７０４へ進む。ステップ７０３では、新規ロック獲得７０３の手続きを実行する。この新規ロック獲得７０３の詳細な内容は図8 を参照して後述する。

【００１９】ステップ７０４，７０５では、一命令でメモリに対する読み書きを不可分に実行できるＴＳ(Test and Set) 命令を使用して、ロック用データ構造体lock__dataのメンバlockの値をtrueにする。なお、ステップ７０５は、他プロセッサが危険領域の命令列を実行している場合には、その危険領域の実行が終了するまで待つ処理である。ＴＳ命令の詳細は「オペレーティング・システムの機能と構成、高橋他、岩波書店、１９８３、１４７頁」に記載されている。ＴＳ命令により、ステップ７０６からステップ７０７の処理2 またはステップ７０９の処理2 までの処理は、ただ一つのプロセッサで実行されることが保証される。

【００２０】ステップ７０６では、ロック用データ構造体lock__dataのメンバanybody の値がNO_PROCESSか否かを判定し、値がNO_PROCESSなら(すなわち、危険領域の命令列5 を他プロセスが実行していないなら) ステップ７０７へ進み、値がNO_PROCESSでないなら(すなわち、危険領域の命令列を他プロセスが実行しているなら) ステップ７０９へ進む。

【００２１】ステップ７０７では、処理1 で、ロック用データ構造体lock__dataのメンバanybody の値をEXIST__PROCESSにする。また、処理2 で、ロック用データ構造体lock__dataのメンバlockの値をFALSE にする。ステップ７０８では、ロック獲得４０４の呼び出し元に戻る。

【００２２】ステップ７０９では、処理1 で、ロック用データ構造体lock__dataのメンバlistに自プロセスを登

7

録し、他プロセスの実行終了を待つ。処理2で、ロック用データ構造体lock_dataのメンバlockの値をFALSEにする。ステップ710では、処理1で、プロセス管理表117の自プロセスの実行状態203を"中断"にする。処理2で、自プロセスを実行していたプロセッサのレジスタの内容を退避し、その退避した領域のアドレスをプロセス管理表117のレジスタ退避領域204に格納する。処理3で、実行中プロセス管理表116から自プロセスのプロセス番号を削除する。ステップ711では、自プロセスを実行していたプロセッサに他の"実行可能"状態にあるプロセスを割り付けるために、プロセス切り替え711の手続きを実行する。このプロセス切り替え711の詳細な内容は図11を参照して後述する。ステップ712では、ロック獲得404の呼び出し元に戻る。

【0023】図8は、新規ロック獲得703の処理を示すPAD図である。ステップ801，802では、一命令でメモリに対する読み書きを不可分に実行できるTS命令を使用して、ロック用データ構造体lock_dataのメンバlockの値をtrueにする。なお、ステップ802は、他プロセッサが危険領域の命令列を実行している場合には、その危険領域の実行が終了するまで待つ処理である。TS命令により、ステップ801からステップ808の処理3またはステップ812の処理3までの処理は、ただ一つのプロセッサで実行されることが保証される。

【0024】ステップ803では、第1引数で指定されたロック用データ構造体lock_dataのメンバprocessorの値が−1か否かを判定し、−1（プロセスがプロセッサに割り付けられていない）ならステップ804に進み、−1でないならステップ807へ進む。ステップ804では、第2引数のフラグflagの値がプロセッサ番号（1～N）か否かを判定し、プロセッサ番号（1～N）ならステップ805へ進み、フラグflagの値がプロセッサ番号でない（−1）ならステップ806へ進む。ステップ805では、フラグflagの値であるプロセッサ番号（1～N）をロック用データ構造体lock_dataのメンバprocessorの値に設定する。そして、ステップ807へ進む。

【0025】ステップ806では、自プロセスが割り付けられているプロセッサ番号を実行中プロセス管理表116から調べて、そのプロセッサ番号をロック用データ構造体lock_dataのメンバprocessorの値に設定する。なお、共有データが現時点で存在しているキャッシュに対応するプロセッサの番号を調べて、そのプロセッサ番号をロック用データ構造体lock_dataのメンバprocessorの値に設定してもよい。そして、ステップ807へ進む。

【0026】ステップ807では、ロック用データ構造体lock_dataのメンバanybodyの値がNO_PROCESSか否か

8

を判定し、値がNO_PROCESSなら（すなわち、危険領域の命令列を他プロセスが実行していないなら）ステップ808へ進み、値がNO_PROCESSでないなら（すなわち、危険領域の命令列を他プロセスが実行しているなら）ステップ812へ進む。

【0027】ステップ808では、処理1で、ロック用データ構造体lock_dataのメンバanybodyの値をEXIST_PROCESSにする。また、処理2で、ロック用データ構造体lock_dataのメンバprocessorの値をプロセス管理表117の割付プロセッサ番号206に設定する。また、処理3で、ロック用データ構造体lock_dataのメンバlockの値をFALSEにする。ステップ809では、ロック用データ構造体lock_dataのメンバprocessorの値と自プロセスが割り付けられているプロセッサ番号とが不一致か否かを判定し、不一致ならステップ810へ進み、一致ならステップ811へ進む。ステップ810では、処理1で、プロセス管理表117の自プロセスの実行状態203を"実行可能"にする。処理2で、自プロセスを実行していたプロセッサのレジスタの内容を退避し、その退避した領域のアドレスをプロセス管理表117のレジスタ退避領域204に格納する。処理3で、実行可能プロセスリスト118に自プロセスのプロセス番号を登録する。処理4で、実行中プロセス管理表116から自プロセスのプロセス番号を削除する。ステップ711では、自プロセスを実行していたプロセッサに他の"実行可能"状態にあるプロセスを割り付けるために、プロセス切り替え711の手続きを実行する。このプロセス切り替え711の詳細な内容は図11を参照して後述する。そして、ステップ811へ進む。ステップ811では、新規ロック獲得703の呼び出し元に戻る。

【0028】ステップ812では、処理1で、ロック用データ構造体lock_dataのメンバlistに自プロセスを登録し、他プロセスの実行終了を待つ。処理2で、ロック用データ構造体lock_dataのメンバprocessorの値をプロセス管理表117の割付プロセッサ番号206に設定する。処理3で、ロック用データ構造体lock_dataのメンバlockの値をFALSEにする。ステップ813では、処理1で、プロセス管理表117の自プロセスの実行状態203を"中断"にする。処理2で、自プロセスを実行していたプロセッサのレジスタの内容を退避し、その退避した領域のアドレスをプロセス管理表117のレジスタ退避領域204に格納する。処理3で、実行中プロセス管理表116から自プロセスのプロセス番号を削除する。ステップ711では、自プロセスを実行していたプロセッサに他の"実行可能"状態にあるプロセスを割り付けるために、プロセス切り替え711の手続きを実行する。このプロセス切り替え711の詳細な内容は図11を参照して後述する。ステップ814では、新規ロック獲得703の呼び出し元に戻る。

【0029】図9は、ロック解放406の処理を示すP

9

AD図である。ステップ９０１では、新規ロック解放９０１の手続きを実行する。この新規ロック解放９１１の詳細な内容は図１０を参照して後述する。ステップ９０２では、引数のロック用データ識別子ｉｄに対応するロック用データ構造体lock__dataを得る。ステップ９０３，９０４では、一命令でメモリに対する読み書きを不可分に実行できるＴＳ命令を使用して、ロック用データ構造体lock__dataのメンバlockの値をtrueにする。なお、ステップ９０４は、危険領域の命令列を実行している場合には、その危険領域の実行が終了するまで待つ処理である。ＴＳ命令により、ステップ９０５からステップ９０６の処理2またはステップ９０８の処理2までの処理は、ただ一つのプロセッサで実行されることが保証される。

【００３０】ステップ９０５では、ロック用データ構造体lock__dataのメンバlistが空か否かを判定し、空なら（すなわち、他プロセスが待機していないなら）ステップ９０６へ進み、空でないなら（すなわち、他プロセスが待機しているなら）ステップ９０８へ進む。

【００３１】ステップ９０６では、処理1で、ロック用データ構造体lock__dataのメンバflagの値にNO_PROCESSを設定する。処理2で、ロック用データ構造体lock__dataのメンバlockの値をFALSEにする。ステップ９０７では、ロック解放４０６の呼び出し元に戻る。

【００３２】ステップ９０８では、処理1で、ロック用データ構造体lock__dataのメンバlistから待機しているプロセスを一つ取り出す。処理2で、ロック用データ構造体lock__dataのメンバlockの値をFALSEにする。ステップ９０９では、処理1で、プロセス管理表１１７の前記取り出したプロセスの実行状態２０３を“中断”から“実行可能”に変更する。処理2で、実行可能プロセスリスト１１８に前記取り出したプロセスのプロセス番号を登録する。処理3で、ロック解放４０６の呼び出し元に戻る。

【００３３】図１０は、新規ロック解放９０１の処理を示すＰＡＤ図である。ステップ９０１ａでは、ロック獲得４０４のステップ８０８の処理2またはステップ８１２の処理2で設定したプロセス管理表１１７の割付プロセッサ番号２０６の値をクリアする。割付プロセッサ番号２０６の値がクリアされると、対応するプロセスを任意のプロセッサに割り付け可能となる。ステップ９０１ｂでは、新規ロック解放９０１の呼び出し元に戻る。

【００３４】図１１は、プロセス切り替え７１１の処理を示すＰＡＤ図である。ステップ１００３では、実行可能プロセスリスト１１８からプロセスを一つ選択する。このとき、実行可能プロセスリストの先頭のプロセスを選択してもよいし、優先度を導入してその優先度の一番高いプロセスを選択してもよい。なお、ステップ１００７から戻ってきてステップ１００３を再実行したときは、先に選択したプロセス以外のプロセスを選択する。

10

【００３５】ステップ１００４では、選択したプロセスのプロセス管理表１１７の割付プロセッサ番号２０６が設定されているか否かを判定し、設定されている場合はステップ１００５へ進み、設定されていない場合はステップ１００９へ進む。ステップ１００５では、設定されている割付プロセッサ番号２０６と自プロセッサ（プロセス切り替え７１１を実行しているプロセッサ）の番号が一致するか否かを判定し、一致した場合にはステップ１００９へ進み、一致しない場合にはステップ１００７へ進む。ステップ１００７では、前記ステップ１００３に戻り、先に選択したプロセス以外のプロセスを選択する。

【００３６】ステップ１００９では、処理1で、前記選択したプロセスを実行可能プロセスリスト１１８から削除する。処理2で、プロセス管理表１１７の前記選択したプロセスの実行状態２０３を“実行可能”から“実行中”に変更する。処理3で、プロセス管理表１１７の前記選択したプロセスのレジスタ退避領域２０４により自プロセッサのレジスタに内容を復帰する。処理4で、前記選択したプロセスの番号を実行中プロセス管理表１１６に登録する。処理5で、プロセス切り替え７１１の呼び出し元に戻る。

【００３７】なお、上記プロセス切り替え７１１は、タイマー割り込み等が発生した時も呼び出される。その時、タイマー割り込み発生を受け取ったプロセッサは、実行中であったプロセスのレジスタ内容を退避し、そのプロセスの実行状態を“実行中”から“実行可能”に変更し、そのプロセスを実行可能プロセスリストに登録した後、プロセス切り替え７１１を呼び出す。

【００３８】以上のマルチプロセッサ計算機システム１０１によれば、共有データにアクセスする命令列５０５をロック獲得４０４とロック解放４０６とで挟むと共にロック獲得４０４の第2引数でプロセッサ１０２(1)～１０２(N)を指定することにより、共有データにアクセスするプロセスを指定したプロセッサに割り付けることが出来る。このため、キャッシュ１０３(1)～１０３(N)間での無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。

【００３９】

【発明の効果】本発明のプロセッサ割付方法およびマルチプロセッサ計算機システムによれば、共有データにアクセスする複数のプロセスを同一のプロセッサに割り付けるため、キャッシュ間の無駄なデータ転送がなくなり、システム全体の実行効率を向上させることができる。

【図面の簡単な説明】

【図1】本発明の一実施形態にかかるマルチプロセッサ計算機システムのブロック図である。

【図2】図1のマルチプロセッサ計算機システムのプロセス管理表の構成図である。

11

【図3】図1のマルチプロセッサ計算機システムの実行中プロセス管理表の構成図である。

【図4】ロック初期化、ロック獲得、ロック解放の各手続きの説明図である。

【図5】ロック初期化、ロック獲得、ロック解放の各手続きを含むプログラム命令列の例示図である。

【図6】ロック初期化手続きの処理を示すＰＡＤ図である。

【図7】ロック獲得手続きの処理を示すＰＡＤ図である。

【図8】新規ロック獲得手続きの処理を示すＰＡＤ図である。

【図9】ロック解放手続きの処理を示すＰＡＤ図である。

【図10】新規ロック解放手続きの処理を示すＰＡＤ図である。

12

【図11】プロセス切り替え手続きの処理を示すＰＡＤ図である。

【符号の説明】

101：マルチプロセッサ計算機システム、102：プロセッサ、103：キャッシュ、104：２次記憶装置、105：システム・バス、106：メインメモリ、107：ロック初期化手続き格納領域、108：ロック獲得手続き格納領域、109：ロック解放手続き格納領域、110：ロック用データ格納領域、111：ロック用変数格納領域、112：ロック獲得フラグ格納領域、113：割付プロセッサ番号格納領域、114：待機プロセスリスト、116：実行中プロセス管理表、117：プロセス管理表、118：実行可能プロセスリスト、119：プロセス切り替え手続き格納領域、402：ロック初期化手続き（命令）、404：ロック獲得手続き（命令）、406：ロック解放手続き（命令）。

【図2】

図2.

プロセス管理表　201　202　203　204　205　117　208

インデックス	プロセス番号	実行状態	レジスタ退避領域	次プロセス・インデックス	割付プロセッサ番号
1	1234	実行中	——	——	——
2	1256	実行可能	r1=0x00001234 …	5	3
⋮					
12	1287	中断	r1=0x00001111 …	END	——
⋮					

【図3】

図3

実行中プロセス管理表　301　302　310

プロセッサ番号	プロセス番号
1	1234
2	256
⋮	
N	87

【図5】

図5

プログラム格納領域　120

プログラム命令列　501

```
program()
{
    init_lock( idp );      402
    ⋮
    lock( id , flag );     404
    共有データへの
    アクセス処理;          505
    unlock( id );          406
    ⋮
}
```

【図4】

図4

機能	手続きの仕様	
	手続き名	手続きの引き数
ロック獲得	lock	1．ロック用データ識別子 id 2．ロック獲得の際、共有データの割り付けられているプロセッサに、プロセスを移動するかどうかを示すフラグ flag
ロック解放	unlock	1．ロック用データ識別子 id
ロック初期化	init_lock	1．ロック用データ識別子を高するための変数格納領域のアドレス idp

404　406　402

【図6】

図6

init_lock(識別子を格納するアドレスidp)　402

ロック用データ構造体を確保
メンバlock……ロックに使用
メンバanybody……実装環境にプロセスが存在することを示すフラグ　601
メンバprocessor…プロセッサ番号
メンバlist……ロックを獲得できなかったプロセスのリスト

ロック用データ構造体を初期化
メンバlock　←　FALSE
メンバanybody　←　NO_PROCESS
メンバprocessor　←　-1
メンバlist　←　空を示す識別子　602

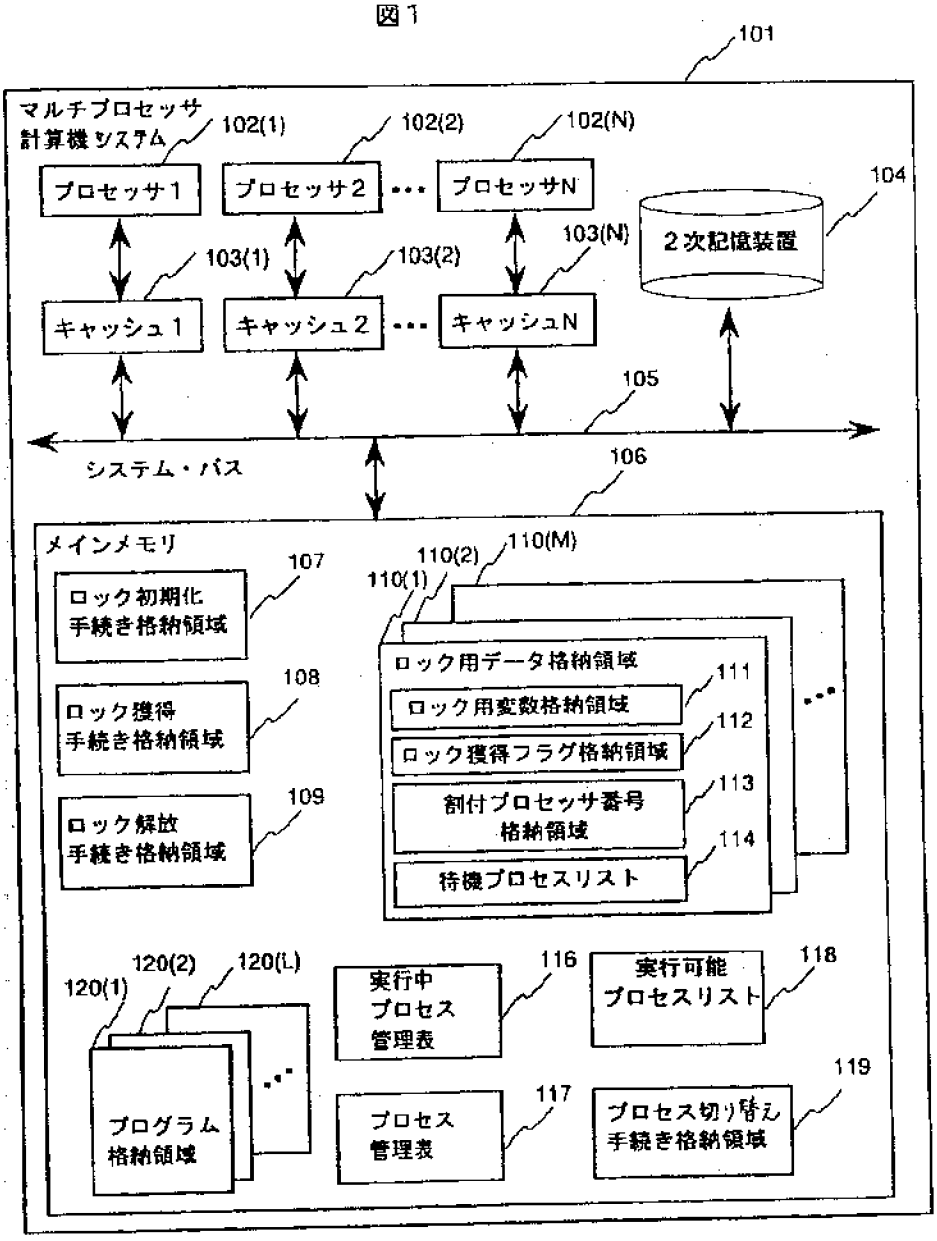
ロック用データ構造体と識別子を対応付ける　603

識別子をアドレスidpに格納する　604

return　605

【 図1 】

図1


101
マルチプロセッサ
計算機システム
102(1)
102(2)
102(N)
プロセッサ1
プロセッサ2
プロセッサN
104
2次記憶装置
103(1)
103(2)
103(N)
キャッシュ1
キャッシュ2
キャッシュN
105
システム・バス
106
メインメモリ
107
ロック初期化
手続き格納領域
108
ロック獲得
手続き格納領域
109
ロック解放
手続き格納領域
110(1)
110(2)
110(M)
ロック用データ格納領域
111
ロック用変数格納領域
112
ロック獲得フラグ格納領域
113
割付プロセッサ番号
格納領域
114
待機プロセスリスト
120(1)
120(2)
120(L)
プログラム
格納領域
116
実行中
プロセス
管理表
118
実行可能
プロセスリスト
117
プロセス
管理表
119
プロセス切り替え
手続き格納領域

【図7】

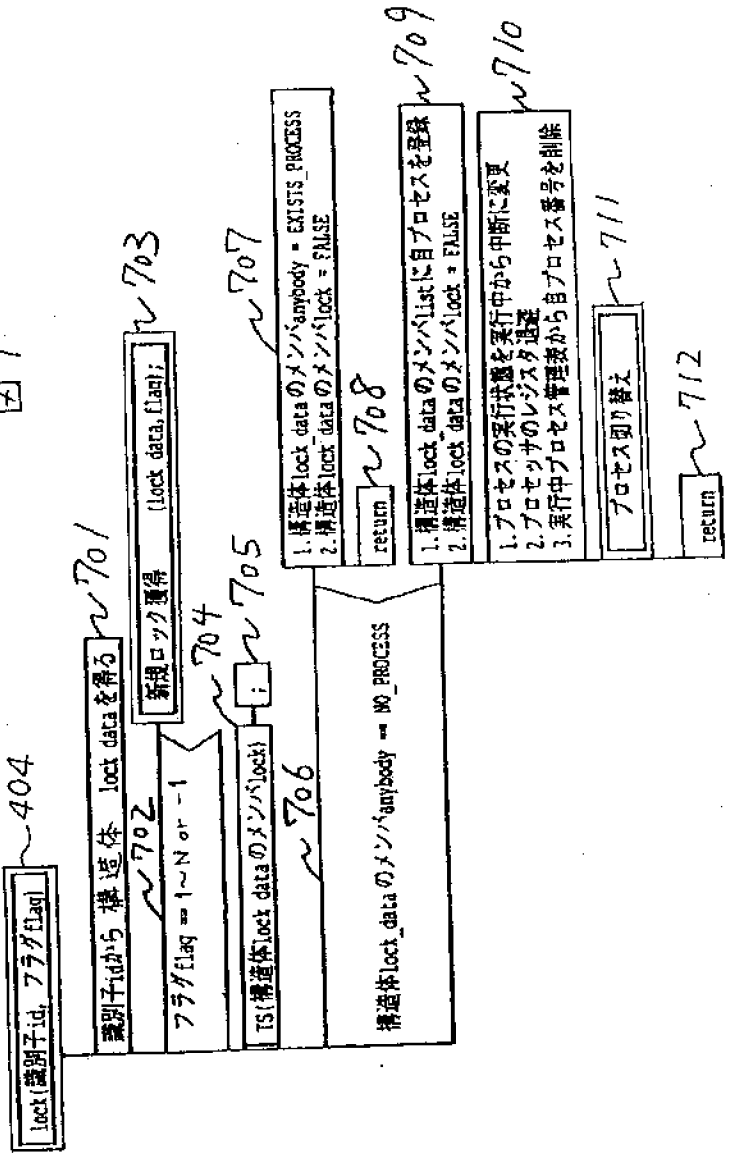


図7
lock(識別子id, フラグflag)
404
識別子idから構造体 lock data を得る
701
新規ロック獲得 (lock data, flag);
703
フラグflag == 1~Nor -1
702
TS(構造体lock_dataのメンバlock)
704
i
705
構造体lock_dataのメンバanybody == NO_PROCESS
706
1.構造体lock_dataのメンバanybody = EXISTS_PROCESS
2.構造体lock_dataのメンバlock = FALSE
707
return
708
1.構造体lock_dataのメンバlistに自プロセスを登録
2.構造体lock_dataのメンバlock = FALSE
709
1.プロセスの実行状態を実行中から中断に変更
2.プロセッサのレジスタ退避
3.実行中プロセス管理表から自プロセス番号を削除
710
プロセス切り替え
711
return
712

【図8】

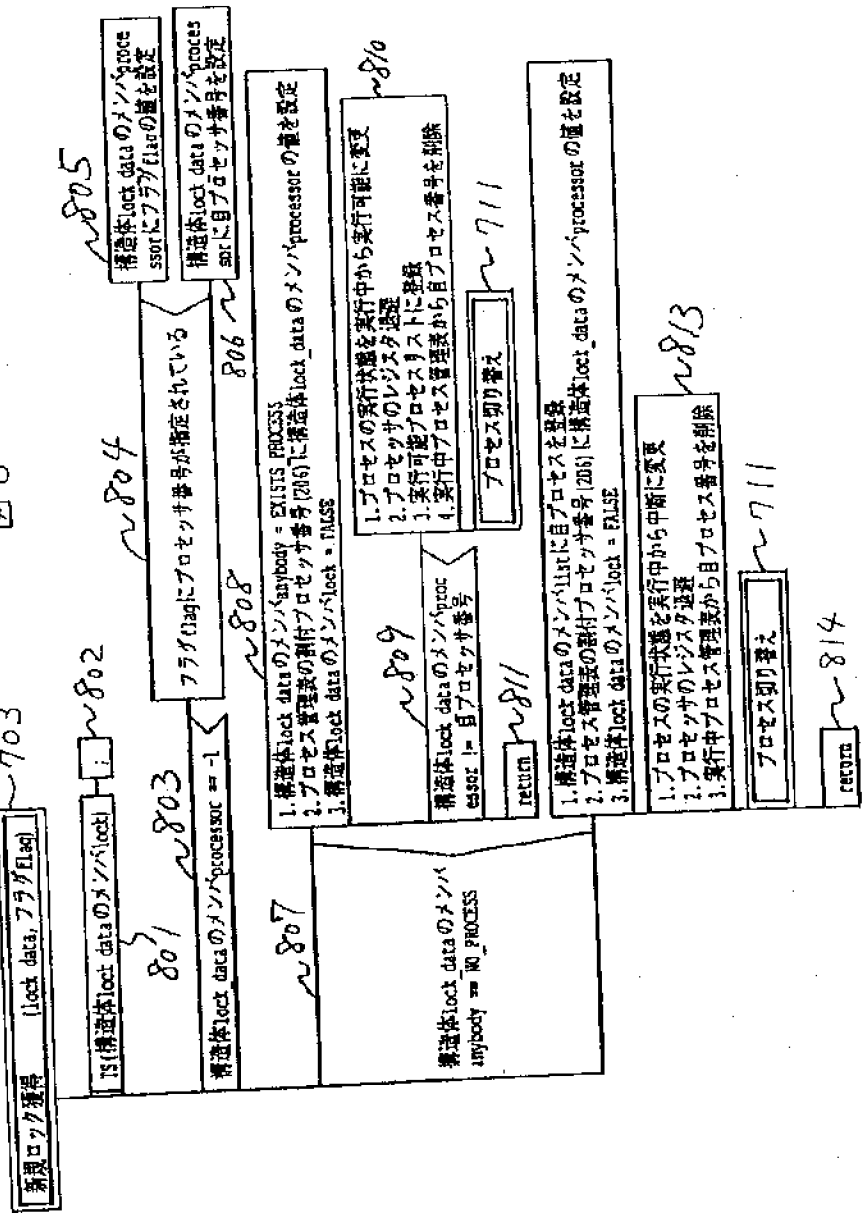
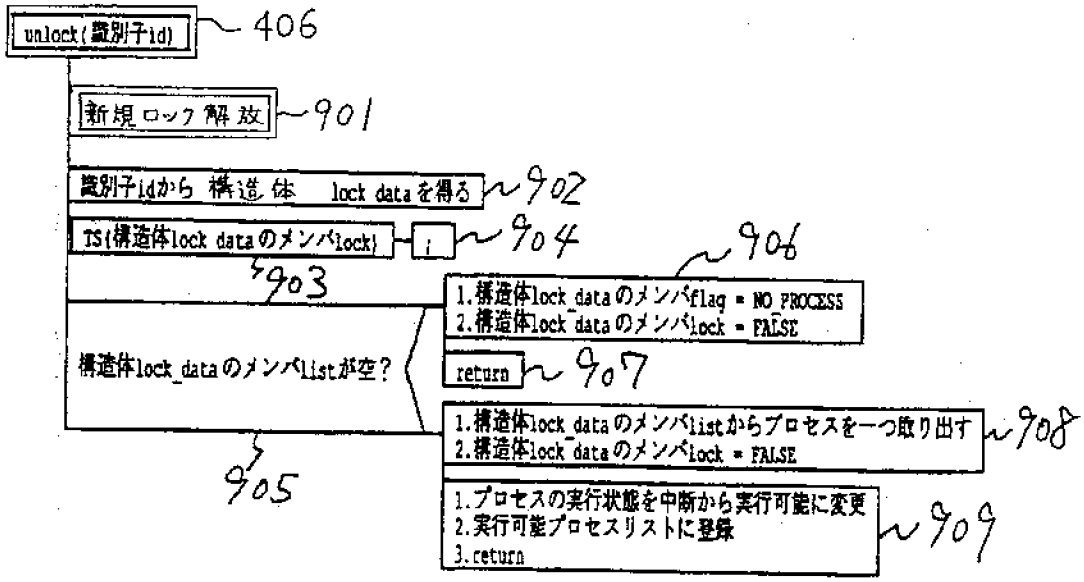

新規ロック獲得 (lock data, フラグflag)
703
TS(構造体lock dataのメンバlock)
802
801
803
構造体lock dataのメンバprocessor == -1
フラグflagにプロセッサ番号が指定されている
804
805
構造体lock dataのメンバprocessorにフラグflagの値を設定
806
構造体lock dataのメンバprocessに自プロセッサ番号を設定
808
1.構造体lock dataのメンバanybody = EXISTS PROCESS
2.プロセス管理表の割付プロセッサ番号(2bit)に構造体lock_dataのメンバprocessorの値を設定
3.構造体lock dataのメンバlock = FALSE
807
構造体lock dataのメンバanybody == NO_PROCESS
809
構造体lock dataのメンバprocessor != 自プロセッサ番号
811
return
910
1.プロセスの実行状態を実行中から実行可能に変更
2.プロセッサのレジスタ退避
3.実行可能プロセスリストに登録
4.実行中プロセス管理表から自プロセス番号を削除
711
プロセス切り替え
1.構造体lock dataのメンバflagに自プロセスを登録
2.プロセス管理表の割付プロセッサ番号(2bit)に構造体lock_dataのメンバprocessorの値を設定
3.構造体lock dataのメンバlock = FALSE
913
1.プロセスの実行状態を実行中から中断に変更
2.プロセッサのレジスタ退避
3.実行中プロセス管理表から自プロセス番号を削除
711
プロセス切り替え
814
return

図8

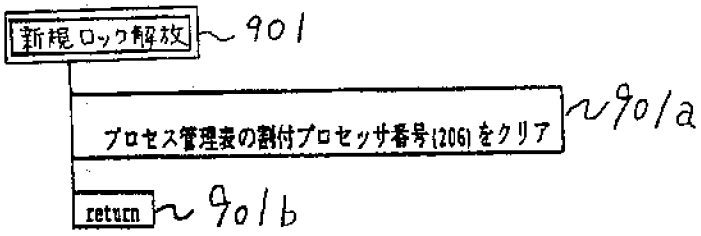
【図9】

図9


unlock(識別子id)
406
新規ロック解放
901
識別子idから構造体 lock dataを得る
902
TS(構造体lock dataのメンバlock)
;
904
903
906
1.構造体lock_dataのメンバflag = NO_PROCESS
2.構造体lock_dataのメンバlock = FALSE
構造体lock_dataのメンバlistが空?
return
907
1.構造体lock_dataのメンバlistからプロセスを一つ取り出す
2.構造体lock_dataのメンバlock = FALSE
908
905
1.プロセスの実行状態を中断から実行可能に変更
2.実行可能プロセスリストに登録
3.return
909

【図10】

図10


新規ロック解放
901
プロセス管理表の割付プロセッサ番号(206)をクリア
901a
return
901b

【 図1 1 】

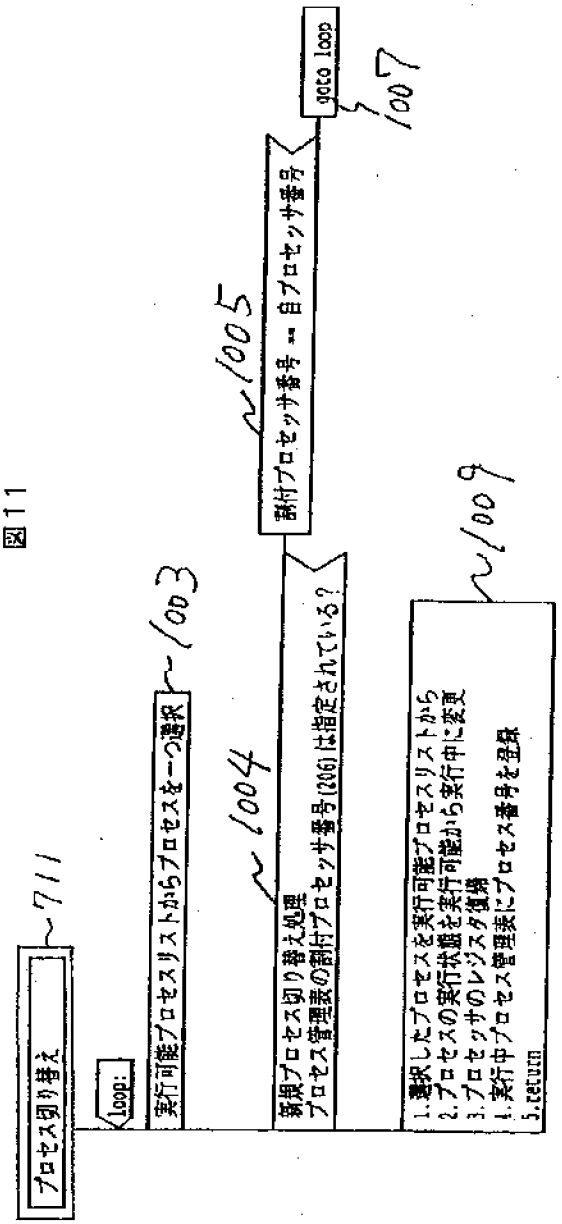


図11

フロントページの続き

(72)発明者　堀川　和雄
神奈川県川崎市幸区鹿島田890番地の12
株式会社日立製作所情報・通信開発本部内

(72)発明者　林　剛久
神奈川県川崎市幸区鹿島田890番地の12
株式会社日立製作所情報・通信開発本部内

(72)発明者　山田　公稔
神奈川県横浜市戸塚区戸塚町5030番地　株式会社日立製作所ソフトウエア開発本部